Lab 02 - Actors

Introduction

While working on this seminar you will learn how to build applications using the *Actor Model* paradigm as introduced in the lecture In this paradigm actors are running concurrently and communicate via message-passing only as it uses a *share nothing* approach, making it very different from typical object-oriented applications. Another outcome of this seminar is to demonstrate how the various interaction patterns are applied in applications.

Exercise

Start: 2025-04-25 Ende: 2024-05-16

The task in this seminar is to implement parts, i.e., a living room with a kitchen, of a *home automation system*. This home automation system contains several sensors measuring the environment and actuators for reacting accordingly. Abbildung 1 illustrates the home automation system and its connected devices. Your implementation has to use an actor-based architecture and we propose using the Akka (https://akka.io/) framework for your implementation.

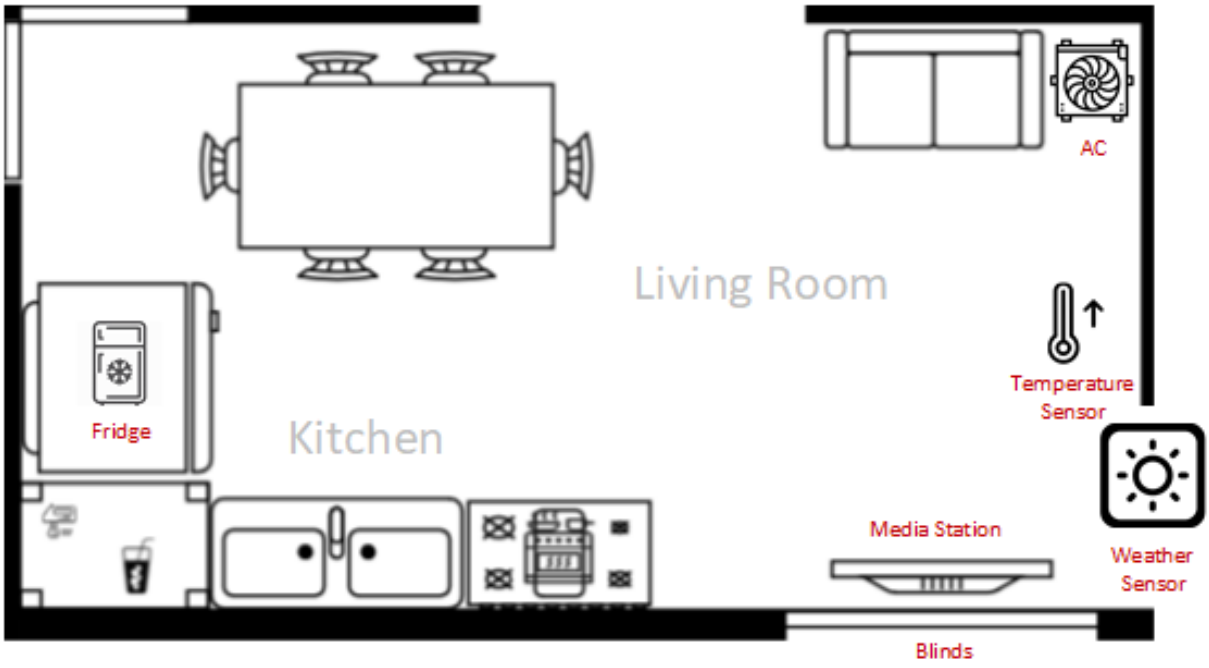

Abbildung 1: Illustration of a home automation system and its connected devices.

Additionally, you need to implement an external actor-based system for processing messages from the home automation system. These two systems are decoupled and communicate through GRPC.

Environment

Functions of devices in the home automation system may depend on environmental conditions, which are not controlled by the system itself, e.g., a heating device regulating the temperature depending on the environmental temperature. In this exercise we don't use any real-world values, but rather simulate the necessary environmental variables by the means of actors and an external source. In general, the environment provides two values: Temperature and weather condition.

1. **Using Actors for Simulating the Environment:** The simulation of the environment has to provide two values, temperature and weather condition, and slightly deviate them from their respective initial values over time. Considering the temperature the simulator should periodically increase or decrease the temperature by a random value, e.g., start with 23°C increase by 0.5, increase by 0.7, decrease by 0.2 and so on. For the weather condition it is sufficient to consider a couple of classes like sunny or cloudy.
   In essence the environment simulator should provide:

   - An Actor that simulates environmental temperature, changing over time.
   - An Actor that simulates weather conditions, changing over time.

2. **Connect to an external Source providing Environment values:** There is a weather simulation server running in the FHV infrastructure that provides values for temperature and weather conditions. This simulation server provides its values through MQTT and can be reached at [10.0.40.161:1883]. Please note that you must be in the FHV VPN or eduroam to access the server.

The system should be able to switch between the internal (actor-based) and external simulation. Additionally, the system needs to be able to set concrete values for both environment variables and deactivate the simulation, i.e., no values from the chosen simulation are being processed.

Devices

The home automation system contains several connected devices. Some of these devices are used passively to measure environmental conditions, i.e., they are sensors, while others are used to actively manipulate and interact with physical real-world components, i.e., they are actuators.

The home automation system contains the following devices:

- *Temperature Sensor* measures the environmental temperature and wraps it in a custom datatype including a unit.
- *Weather Sensor* measures the weather condition.
- *AC (Air Conditioning)* regulates the AC depending on the measured temperature.
- *Blinds* regulates the blinds depending on the measured weather condition, e.g., closes the blinds if the weather is sunny.
- *Media Station* allows playing movies.
- *Fridge* manages products and allows ordering new products. Based on the currently contained products an order might no be realizable.

The Fridge is a special kind of device in our system as it contains itself two additional sensors, measuring weight of and space taken by contained products:

- The Fridge has a maximum number of storable products.
- The Fridge has a maximum weight load.
- Each Product has a price and a weight.

The fridge can also be used to automatically order products from an external system. In this case, the fridge contacts the external system through GRPC and sends the order to it. The order is then processed there and a receipt is returned.

Functionality and Rules

The home automation system provides the following functionality:

- Users can consume products from the Fridge.
- Users can order products at the Fridge.
- A successful order returns a receipt.
- The Fridge allows for querying the currently stored products.
- The Fridge allows for querying the history of orders.
- Users can play movies at the media station.

The home automation system has to adhere to the following rules:

- If the weather is sunny the blinds will close.
- If the weather is not sunny the blinds will open (unless a movie is playing).
- If a movie is playing the blinds are closed.
- A new movie cannot be started if another movie is already playing.
- If the temperature is above 20°C the AC starts cooling.
- If the temperature is below 20°C the AC turns off.
- The Fridge can only process an order if there is enough room in the fridge, i.e., the contained products and newly order products do not exceed the maximum number of storable products.
- The Fridge can only process an order if the weight of the sum of the contained products and newly order products does not exceed its maximum weight capacity.
- If a product runs out in the fridge it is automatically ordered again.

External Order-Processor System

The external system is used to process orders from the fridge. It receives a product and an amount and processes the order. After successfully handling the order (all orders can be processed) a receipt is returned with the necessary values (items, amount, price…). This external system is also realized using an actor-based architecture in akka. However, the actor for processing the order should not be part of the home automation system (and it is different from the *OrderProcessor* in the fridge), but rather placed in a separate system. Hence, the two systems should be decoupled and be able to be deployed independently. To accomplish this, communication through a protocol is necessary. In this example, we will utilize gRPC (***g****RPC* ***R****emote* ***P****rocedure* ***C****alls)* as Akka already provides support for it (https://doc.akka.io/libraries/akka-grpc/current/index.html). There is an example, video tutorial and github repository available (https://doc.akka.io/libraries/akka-grpc/current/quickstart-java/index.html). There is no need to implement a full micro-service based approach, but if you prefer you can use this approach as well (https://doc.akka.io/libraries/guide/microservices-tutorial/index.html).

Implementation

Use the Akka framework to implement the functionality described above and provide a command line interface (REPL) or web interface for interacting with the home automation system. The interface should provide the means to execute all functionality described above as well as manipulating the environment variables. While designing your architecture consider using a hierarchical setup of Actors and use fitting interaction patterns (some hints are given below). You can find a documentation for Akka here (https://doc.akka.io/docs/akka/current/typed/index.html), for the interaction patterns here (https://doc.akka.io/docs/akka/current/typed/interaction-patterns.html), overview for hierarchies in actors (https://doc.akka.io/docs/akka/current/typed/guide/tutorial_1.html) and for guardians as well as lifecycles (https://doc.akka.io/docs/akka/current/typed/actor-lifecycle.html). A central registration unit for devices is not necessary, however, you may implement one for easier retrieval of actor references. You may consider using the *Blackboard Pattern* for implementing this exercise.

When implementing the communication between the two systems use gRPC as described above. Akka also provides other possibilities for HTTP and REST, which can be used in addition (if necessary).

For implementing the connection to the external weather simulation server and processing its values, you need to implement an MQTT client. The Eclipse Paho project provides an easy to use implementation (see https://www.baeldung.com/java-mqtt-client and https://eclipse.dev/paho/). Akka provides a framework fro stream-processing of values through Alpakka (https://doc.akka.io/libraries/alpakka/current/index.html). This essentially enables you to directly access values from a source and bind it to an actor. There is also support for MQTT (https://doc.akka.io/libraries/alpakka/current/mqtt-streaming.html). A more simplified example can be found here: https://github.com/marcelo-s/akka-alpakka-mqtt/tree/master

Implementation details and hints:

- The actors of the environment simulator are scheduled periodically (see *Scheduling messages to self*).
- The sensors measure the values of the environment (see *Request-Response with ask between two actors* or the environment pushes the values).
- The sensors wrap the raw values from the environment for the actuators (including units).
- The AC and Blinds actuators receive the values from the sensors (see *Fire and Forget*).
- The user queries the fridge (see *Request-Response*).
- The user consumes a product from the fridge (see *Ignoring replies*).
- The user orders products. In this case the fridge should relay this request to a separate *OrderProcessor* actor (see *Per session child Actor*).
- Possibly one actor that handles the UI.

Submission

In this exercise you can work in teams (groups of 2) and the submission must be made by May 16th, 2025.

Submit a zip file containing the following artifacts:

- Source-Code of your implementation as gradle project

- Documentation of the chosen architecture. The documentation should explain design decisions, interaction patterns between actors and give an overall view of the architecture (e.g., class diagram)
- Readme on how to run and interact the with the application.